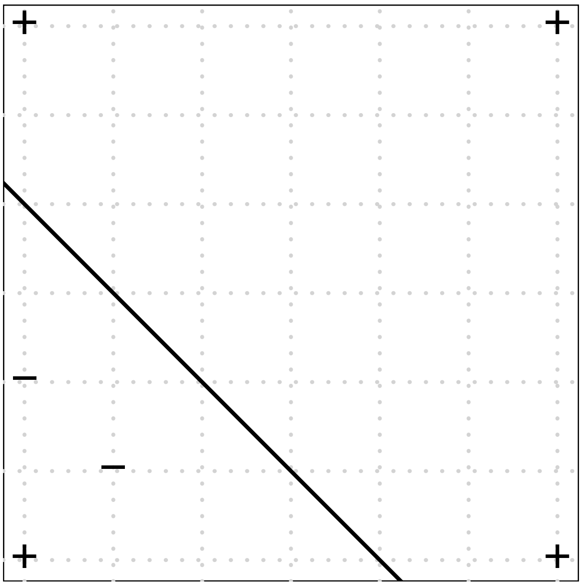
Solution 1: Soft Margin Classifier

(a) The hyperplane is given by:

$$\theta_1 x_1^{(i)} + \theta_2 x_2^{(i)} + \theta_0 = 0 \tag{1}$$

Plugging in the values for the θs and solving for x_2, we get the decision boundary as function of x_1:

$$x_2 = -x_1 + 2 \tag{2}$$

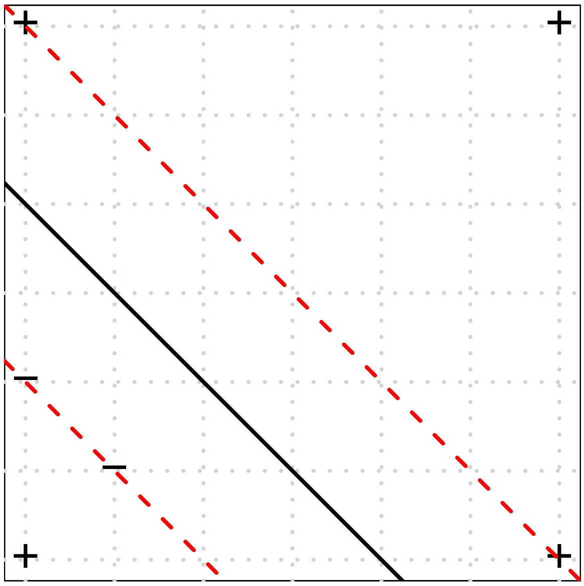


(b) To determine which points are on the margin hyperplanes, we will use the constraint:

$$\zeta^{(i)} = \max\left(0, 1 - y^{(i)}\left(x^{(i)}\hat{\theta} + \hat{\theta}_0\right)\right) \tag{3}$$

$$\begin{cases} (0,0): \ 1 - 1\,(0+0-2) = 3 > 0 \longrightarrow \zeta^{(1)} = 3 \\ (0.5,0.5): \ 1 - (-1)\cdot(0.5+0.5-2) = 0 \longrightarrow \zeta^{(2)} = 0 \\ (0,1): \ 1 - (-1)\cdot(0+1-2) = 0 \longrightarrow \zeta^{(3)} = 0 \\ (0,3): \ 1 - 1\,(0+3-2) = 0 \longrightarrow \zeta^{(4)} = 0 \\ (3,0): \ 1 - 1\,(3+0-2) = 0 \longrightarrow \zeta^{(5)} = 0 \\ (3,3): \ 1 - 1\,(3+3-2) = -3 < 0 \longrightarrow \zeta^{(6)} = 0 \end{cases} \tag{4}$$

$(0.5, 0.5), (0, 1), (0, 3), (3, 0)$ lie on the margin hyperplanes $\zeta^{(i)} = 0$.



Solution 2: Optimization

- Implementation of the PEGASOS algorithm:

```
#' @param y outcome vector
#' @param X design matrix (including a column of 1s for the intercept)
#' @param nr_iter number of iterations for the algorithm
#' @param theta starting values for thetas
#' @param lambda penalty parameter
#' @param alpha step size for weight decay
pegasos_linear <- function(
  y,
  X,
  nr_iter = 50000,
  theta = rnorm(ncol(X)),
  lambda = 1,
  alpha = 0.01)
{

  t <- 1
  n <- NROW(y)

  while(t <= nr_iter){

    f_current = X%*%theta
    i <- sample(1:n, 1)

    # update
    theta <- (1 - lambda * alpha) * theta
    # add second term if within margin
    if(y[i]*f_current[i] < 1) theta <- theta + alpha * y[i]*X[i,]

    t <- t + 1

  }
```

```
  return(theta)

}
```

- Check on a simple example

```
## Check on a simple example
## --------------------------------------------

set.seed(2L)

C = 1

library(mlbench)
library(kernlab)
data = mlbench.twonorm(n = 100, d = 2)


data = as.data.frame(data)
X = as.matrix(data[, 1:2])
y = data$classes
par(mar = c(5,4,4,6))
plot(x = data$x.1, y = data$x.2, pch = ifelse(data$classes == 1, "-", "+"), col = "black",
     xlab = "x1", ylab = "x2")

# recode y
y = ifelse(y == "2", 1, -1)
mod_pegasos = pegasos_linear(y, cbind(1,X), lambda = 1/(C*NROW(y)))

# Add estimated decision boundary:
abline(a = - mod_pegasos[1] / mod_pegasos[3],
       b = - mod_pegasos[2] / mod_pegasos[3], col = "#D55E00")

# Compare to logistic regression:
mod_logreg = glm(classes ~ ., data = data, family = binomial())
abline(a = - coef(mod_logreg)[1] / coef(mod_logreg)[3],
       b = - coef(mod_logreg)[2] / coef(mod_logreg)[3], col =  "#56B4E9",
       lty = 3, lwd = 2)

# decision values
f_pegasos = cbind(1,X) %*% mod_pegasos

# How many wrong classified examples?
table(sign(f_pegasos * y))

##
## -1  1
##  5 95

## compare to kernlab. we CANNOT expect a PERFECT match
## ---------------------------------------------------------------

mod_kernlab = ksvm(classes~.,
                   data = data,
                   kernel = "vanilladot",
                   C = C,
                   kpar = list(),
                   scaled = FALSE)
```

```r
f_kernlab = predict(mod_kernlab, newdata = data, type = "decision")
# How many wrong classified examples?
table(sign(f_kernlab * y))

##
## -1  1
##  5 95

# compare outputs
print(range(abs(f_kernlab - f_pegasos)))

## [1] 0.00014996 0.38049736

# compare coeffs
# (b is negative offset)
all_params = rbind(
  mod_pegasos,
  mod_kernlab = c(-mod_kernlab@b,
  (params <- colSums(X[mod_kernlab@SVindex, ] *
                       mod_kernlab@alpha[[1]] *
                       y[mod_kernlab@SVindex])))
)
all_params

##                                x.1        x.2
## mod_pegasos -0.05743352 -1.347267 -0.7917586
## mod_kernlab -0.09763532 -1.263707 -0.7747026

# seems we were reasonably close

# compare empirical risks
emp_risk <- function(theta){
  f = cbind(1, X) %*% theta
  return(0.5 * sum(theta[2:3]^2) + C*sum(ifelse(f > 0, f, 0)))
}
apply(all_params, 1, emp_risk)

## mod_pegasos mod_kernlab
##    153.3287    143.8169

params = all_params[2,]
# recompute margin
margin = 1 / sqrt(sum(params[2:3]^2))

# compute value of intercept shift (the margin shift is in orthogonal direction
#   to the decision boundary, so this has to be transformed first)
m = - params[2] / params[3]
t_0 = margin / (cos(atan(m)))

# add margins to visualization:
#
abline(a =  -params[1] / params[3],
       b = m, col = "#0072B2")
abline(a =  -params[1] / params[3] + t_0,
       b = m, col = "#0072B2", lty = 2)
abline(a =  -params[1] / params[3] - t_0,
       b = m, col = "#0072B2", lty = 2)
```

```r
# find support vectors on decision border
nv_sv_idx = y[mod_kernlab@SVindex] * f_kernlab[mod_kernlab@SVindex] > 0.98
points(X[mod_kernlab@SVindex[nv_sv_idx],])


# add legends
legend(par('usr')[2], par('usr')[4], , bty='n', xpd=NA, legend=c("1","2"),
       pch=c("-","+"), title="Classes", cex = 0.8)
legend(par('usr')[2], 1.8, , bty='n', xpd=NA,
       legend=c("Pegasos","Logistic","Kernlab","Margin"),
       lty=c(1,3,1,2),
       col = c("#D55E00","#56B4E9","#0072B2","#0072B2"),
       title="", cex = 0.8, lwd = c(1,2,1,1))
```